

| | | | | | | | | |
|--------|--------|-------------------|---------------|---------------|---------------|-----|-----|-----------|
| MMM | MMM | AAAAAAAAA | CCCCCCCCCCCCC | RRRRRRRRRRRRR | 000000000 | | | |
| MMM | MMM | AAAAAAAAA | CCCCCCCCCCCCC | RRRRRRRRRRRRR | 000000000 | | | |
| MMM | MMM | AAAAAAAAA | CCCCCCCCCCCCC | RRRRRRRRRRRRR | 000000000 | | | |
| MMMMMM | MMMMMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMMMMM | MMMMMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMMMMM | MMMMMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRRRRRRRRRRRR | | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRRRRRRRRRRRR | | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRRRRRRRRRRRR | | 000 | 000 |
| MMM | MMM | AAAAAAAAAAAAAAAAA | CCC | RRR | RRR | | 000 | 000 |
| MMM | MMM | AAAAAAAAAAAAAAAAA | CCC | RRR | RRR | | 000 | 000 |
| MMM | MMM | AAAAAAAAAAAAAAAAA | CCC | RRR | RRR | | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMM | MMM | AAA | AAA | CCC | RRR | RRR | 000 | 000 |
| MMM | MMM | AAA | AAA | CCCCCCCCCCCCC | RRR | RRR | | 000000000 |
| MMM | MMM | AAA | AAA | CCCCCCCCCCCCC | RRR | RRR | | 000000000 |
| MMM | MMM | AAA | AAA | CCCCCCCCCCCCC | RRR | RRR | | 000000000 |

```
SSSSSSSS  CCCCCCCC  AAAAAA  NN  NN  EEEEEEEEE  RRRRRRRR
SSSSSSSS  CCCCCCCC  AAAAAA  NN  NN  EEEEEEEEE  RRRRRRRR
SS  SS  CC  CC  AA  AA  NN  NN  EE  RR  RR
SS  SS  CC  CC  AA  AA  NN  NN  EE  RR  RR
SS  SS  CC  CC  AA  AA  NN  NN  EE  RR  RR
SSSSSS  CC  CC  AA  AA  NN  NN  EE  RRRRRRRR
SSSSSS  CC  CC  AA  AA  NN  NN  EE  RRRRRRRR
SS  SS  CC  CC  AAAAAAAAAA  NN  NNNN  EE  RR  RR
SS  SS  CC  CC  AAAAAAAAAA  NN  NNNN  EE  RR  RR
SS  SS  CC  CC  AA  AA  NN  NN  EE  RR  RR
SSSSSSSS  CCCCCCCC  AA  AA  NN  NN  EEEEEEEEE  RR  RR
SSSSSSSS  CCCCCCCC  AA  AA  NN  NN  EEEEEEEEE  RR  RR
```

```
LL  LL  SSSSSSSS
LL  LL  SSSSSSSS
LL  LL  SS
LL  LL  SS
LL  LL  SS
LL  LL  SSSSSS
LL  LL  SSSSSS
LL  LL  SS
LL  LL  SS
LL  LL  SS
LLLLLLLLLL  IIIIIII  SSSSSSSS
LLLLLLLLLL  IIIIIII  SSSSSSSS
```

| | | |
|------|-----|----------------------------------------------------|
| (2) | 79 | DECLARATIONS |
| (3) | 125 | MAC\$SYMBOL SCAN NEXT SYMBOL |
| (4) | 198 | MAC\$CHKREG IS SCANNED SYMBOL A REGISTER |
| (4) | 220 | MAC\$SYMMUM LETTERS A-F CAN START SYMBOL OR NUMBER |
| (4) | 232 | MAC\$XSYMBL NEXT CHARACTER CAN BE AN UPARROW |
| (4) | 253 | MAC\$NUMBER ACCUMULATE NUMBER |
| (5) | 269 | MAC\$SYMSCN SCAN FOR SYMBOL NAME |
| (5) | 304 | MAC\$LCCLSKIP SCAN FOR LOCAL LABEL |
| (5) | 329 | CONVERT SYMBOL TO UPPER CASE FOR TABLE SEARCH |
| (6) | 356 | MAC\$GETSYM ACCUMULATE SYMBOL NAME |
| (7) | 437 | MAC\$XUPARROW CHECK CHARACTER AFTER "" |
| (7) | 462 | MAC\$XPOUND A POUND SIGN WAS DETECTED |
| (8) | 494 | INSERT/SEARCH USER SYMBOL TABLE |
| (9) | 650 | MAC\$REGIS SEE IF THE SCANNED SYMBOL IS A REGISTER |
| (11) | 750 | MAC\$DNUMBER ACCUMULATE DECIMAL NUMBER |
| (12) | 835 | MAC\$GOTLOCLAB PROCESS LOCAL LABEL |
| (13) | 866 | ACCUMULATE BINARY/OCTAL/HEX NUMBERS |


```
0000 1      .TITLE  MAC$SCANER SCANNING ROUTINES
0000 2      .IDENT  'V04-000'
0000 3
0000 4
0000 5 *****
0000 6 *****
0000 7 *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 *  ALL RIGHTS RESERVED.
0000 10
0000 11 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 *  TRANSFERRED.
0000 17
0000 18 *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 *  CORPORATION.
0000 21
0000 22 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24
0000 25 *****
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 : FACILITY:      VAX MACRO ASSEMBLER OBJECT LIBRARY
0000 31
0000 32 : ABSTRACT:
0000 33
0000 34 : The VAX-11 MACRO assembler translates MACRO-32 source code into object
0000 35 : modules for input to the VAX-11 LINKER.
0000 36
0000 37 : ENVIRONMENT:  USER MODE
0000 38
0000 39 : AUTHOR: Benn Schreiber, CREATION DATE: 29-AUG-78
0000 40
0000 41 : MODIFIED BY:
0000 42
0000 43 :      V03-001 MTR0023      Mike Rhodes      01-Feb-1983
0000 44 :      Fix truncation errors in MAC$XSYMBL and MAC$XUPARROW.
0000 45
0000 46 :      V02.16 PCG0008      Peter George      26-Aug-1981
0000 47 :      Fix handling of floating point literals.
0000 48
0000 49 :      V02.15 CNH0040      Chris Hume      15-Oct-1980
0000 50 :      .ENDC ignored after local label in conditional suppressed
0000 51 :      code. (ACTIF.MAR 02.06)
0000 52
0000 53 :      V02.14 HJ0001      Herb Jacobs      14-Aug-1980
0000 54 :      Performance improvement to symbol table search
0000 55
0000 56 :      V01.13 RN0023      R. Newland      3-Nov-1979
0000 57 :      New message codes to get error messages from system
```

0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :--

message file.

V01.12 RN0014 R. Newland 12-Oct-1979
Support for G_floating, H_floating and Octaword data types

V01.12 RN0015 R. Newland 11-Oct-1979
Fix problem with top 32 bits of quadword number.
SPR 11-26479

V01.11 RN0008 R. Newland 29-Aug-1979
31 character symbols

V01.10 RN0005 R. Newland 14-Aug-1979
Variable symbol storage and remove .ALIGN LONG
and .DEBUG statements

V01.09 008 B. Schreiber 22-JAN-1979
Better bookkeeping of pages allocated so they can
be deallocated on multiple assemblies.

```
0000 79      .SBTTL  DECLARATIONS
0000 80      :
0000 81      : INCLUDE FILES:
0000 82      :
0000 83      :
0000 84      :
0000 85      : MACROS:
0000 86      :
0000 87      :
0000 88      $MAC_GENVALDEF      ;USEFUL SYMBOLS
0000 89      $MAC_SYMBLKDEF     ;SYMBOL BLOCK DEFINITIONS
0000 90      $MAC_GRAMMARDEF    ;TERMINAL GRAMMAR SYMBOLS
0000 91      $MAC_INTCODEF     ;INTERMEDIATE CODE DEFINITIONS
0000 92      $MAC_CTLFLGDEF    ;CONTROL FLAG DEFINITIONS
0000 93      $MAC_CRFLAGDEF    ;DEFINE CREF CONTROL FLAGS
0000 94      $MAC_OPRDEF       ;DEFINE OPERAND DESCRIPTOR BITS
0000 95      $MACMSGDEF        ; Define message codes
0000 96      :
0000 97      :
0000 98      : EQUATED SYMBOLS:
0000 99      :
0000 100     :
0000 101     :
0000 102     : OWN STORAGE:
0000 103     :
0000 104     :
00000000 105     .PSECT  MAC$RO_DATA,NOEXE,NOWRT,GBL, LONG
0000 106     :
0000 107     :
0000 108     TWO_CHR_REG_NAM:      ;TABLE OF NAMES FOR TWO CHAR REGISTERS
0000 109     .ASCII  /R0R1R2R3R4R5R6R7R8R9SPPCAPFPIVDV/
35 52 34 52 33 52 32 52 31 52 30 52 0000
43 50 50 53 39 52 38 52 37 52 36 52 000C
    56 44 56 49 50 46 50 41 0018
    00000020 0020 110 TWO_CHR_REG_SIZ=-TWO_CHR_REG_NAM      ;SIZE OF TABLE
    0020 111 TWO_CHR_REG_VAL:      ;VALUES ASSOCIATED WITH ABOVE REGS
OF 0E 09 08 07 06 05 04 03 02 01 00 0020 112     .BYTE  0,1,2,3,4,5,6,7,8,9,14,15,12,13,14,15
    OF 0E 0D 0C 002C
    0030 113
00000030 114     .PSECT  MAC$RO_DATA,NOWRT,NOEXE,GBL, LONG
    0030 115 MAC$AL_RGNM_TAB::      ;POINTER TO REGISTER NAME SYMBOL
    0030 116      ;BLOCKS (USED FOR CREF)
    0030 117     .IRP    FOO,<R0,R1,R2,R3,R4,R5,R6,R7,R8,R9,R10,R11,AP,FP,SP,PC>
    0030 118     .PSECT  MAC$RO_DATA,NOWRT,NOEXE,GBL, LONG
    0030 119     .LONG   REG_SYMB 'FOO
    0030 120     .PSECT  MAC$RW_DATA,WRT,NOEXE, LONG
    0030 121     $MAC_INSERT_SYM 'FOO,,,REG_SYMB_'FOO
    0030 122     $MAC_INSERT_SYM 'FOO
00000030 123     .ENDM
```



```
0204 125 .SBTTL MAC$SYMBOL SCAN NEXT SYMBOL
0204 126
0204 127 :++
0204 128 :FUNCTIONAL DESCRIPTION:
0204 129 :
0204 130 :THIS ROUTINE ACCUMULATES A SYMBOL NAME IN 'MAC$AB TMP$SYM'.
0204 131 :THE SYMBOL IS THEN LOOKED UP IN THE APPROPRIATE SYMBOL
0204 132 :TABLES (BASED ON THE POSITION IN THE LINE). THE TOKEN TYPE
0204 133 :IS RETURNED IN R8 AND THE POINTER TO THE SYMBOL BLOCK
0204 134 :IS RETURNED IN 'MAC$GL_VALUE'.
0204 135 :
0204 136 :--
0204 137
00000000 138 .PSECT MAC$RO_CODE_P1,NOWRT,GBL, LONG
0000 139
0000 140 MAC$SYMBOL::
21 6B 00EF 30 0000 141 BSBW MAC$SYMSCNUP ;ACCUMULATE SYMBOL NAME
01 E1 0003 142 BBC #FLG$V BOL,(R11),100$ ;BR IF NOT AT BEGINNING OF LINE
3A FFF6' 30 0007 143 BSBW MAC$SKIPSP ;SKIP OVER SPACES
5A 91 000A 144 CMPB R10,#^A/;/ ;DID WE FIND A LABEL?
04 12 000D 145 BNEQ 10$ ;IF NEQ NO
64 10 000F 146 BSBB MAC$CHKREG ;YES--RETURN TOKEN TYPE IN R8
5C 11 0011 147 BRB 200$ ;GO RETURN
0013 148 :
0013 149 :NO LONGER AT BEGINNING OF LINE--SEE IF ASSIGNMENT
0013 150
6B 02 CA 0013 151 10$: BICL2 #FLG$M BOL,(R11) ;DO NOT ALLOW ANY MORE LABELS
3D 5A 91 0016 152 CMPB R10,#^A/=/ ;ASSIGNMENT STATEMENT?
0D 12 0019 153 BNEQ 100$ ;IF NEQ NO
58 10 001B 154 BSBB MAC$CHKREG ;YES--RETURN TOKEN TYPE IN R8
0C 58 91 001D 155 CMPB R8,#ID ;WAS IT AN ID?
4D 12 0020 156 BNEQ 200$ ;IF NEQ GO EXIT
49 6B 0D E3 0022 157 BBCS #FLG$V_OPRND,(R11),200$ ;YES--NOW IN OPERAND FIELD
47 11 0026 158 BRB 200$
0028 159 :
0028 160 :IN OPERAND FIELD?
0028 161
04 6B 0D E1 0028 162 100$: BBC #FLG$V_OPRND,(R11),110$ ;BR IF NOT IN OPERAND FIELD
47 10 002C 163 BSBB MAC$CHKREG ;RETURN TOKEN TYPE IN R8
3F 11 002E 164 BRB 200$
53 0000'CF 9E 0030 165 110$: MOVAB W^MAC$AL_UMCHSHTB,R3 ;Look in user macro table
020F 30 0035 166 BSBW MAC$SRC$SYMTAB
05 50 E9 0038 167 BLBC R0,120$ ;IF LBC THEN NOT FOUND
58 0D 9A 003B 168 MOVZBL #MACTXT,R8 ;FOUND--RETURN TOKEN FOR MACRO
2F 11 003E 169 BRB 200$
0040 170 :
0040 171 :NOT A MACRO--LOOK THROUGH PERMANENT SYMBOL TABLE
0040 172
55 0000'CF D0 0040 173 120$: MOVL W^MAC$GL_OPCLSTPT,R5 ;GET POINTER TO USER-DEFINED OPCODES
06 13 0045 174 BEQL 122$ ;IF EQL NONE DEFINED
01F0 30 0047 175 BSBW MAC$SRC_LIST ;SEE IF USER-DEFINED OPCODE
0D 50 E8 004A 176 BLBS R0,124$ ;IF LBS YES--GO PROCESS IT
53 00000000'EF 9E 004D 177 122$: MOVAB MAC$AL_PRMHSHTB,R3 ;NO--LOOK IN PERMANENT SYMBOL TABLE
01F0 30 0054 178 BSBW MAC$SRC$SYMTAB
12 50 E9 0057 179 BLBC R0,130$ ;IF LBC THEN NOT FOUND
58 0B A1 9A 005A 180 124$: MOVZBL SYM$B_TOKEN(R1),R8 ;PICK UP TOKEN FOR SYMBOL
0E 58 91 005E 181 CMPB R8,#D0PCODE ;WAS THIS AN OPCODE?
```

```
03 13 0061 182 BEQL 125$ ;IF EQL YES
FF9A' 30 0063 183 BSBW MAC$CREF_DIR ;NO--DIRECTIVE--CREF IT IF CREFFING
05 6B 0D E3 0066 184 ; DIRECTIVES.
03 11 0066 185 125$: BBCS #FLG$V_OPRND,(R11),200$ ;NOW IN OPERAND FIELD
006A 186 200$
006C 187 ;
006C 188 ; NOT FOUND--TRY AN IMPLICIT .MCALL
006C 189 ;
FF91' 30 006C 190 130$: BSBW MAC$IMPLMCALL ;TRY AN IMPLICIT MCALL
006F 191 **: BRB 200$ ;RETURN WITH TOKEN
006F 192 ;
006F 193 ; SET POINTER TO SYMBOL BLOCK IN MAC$GL_VALUE
006F 194 ;
0000'CF 51 D0 006F 195 200$: MOVL R1,W*MAC$GL_VALUE ;RETURN POINTER TO SYMBOL BLOCK
05 0074 196 220$: RSB
```



```
0075 198 .SBTTL MAC$CHKREG IS SCANNED SYMBOL A REGISTER
0075 199
0075 200 :++
0075 201 : FUNCTIONAL DESCRIPTION:
0075 202 :
0075 203 : THIS ROUTINE CHECKS TO SEE IF THE SYMBOL SCANNED BY MAC$GETSYM
0075 204 : IS A REGISTER NAME, AND IF NOT, SEARCHES FOR IT IN THE USER'S
0075 205 : SYMBOL TABLE. IF THE SYMBOL IS NOT IN THE USER'S SYMBOL TABLE
0075 206 : IT IS INSERTED. IN ANY CASE, THE TOKEN VALUE RETURNED IN R8
0075 207 : IS EITHER 'ID' OR 'RRREG'. IF 'RRREG', THE REGISTER VALUE
0075 208 : WILL BE RETURNED IN R1. IF 'ID', THE SYMBOL BLOCK ADDRESS
0075 209 : WILL BE RETURNED IN R1.
0075 210 :--
0075 211
0075 212 MAC$CHKREG:
0075 213 BSBW MAC$REGIS ;SEE IF SYMBOL IS A REGISTER NAME
0075 214 TSTL R8 ;IS IT A REGISTER?
0075 215 BNEQ 10$ ;IF NEQ YES
0075 216 BSBW MAC$INSUSRSYMTB ;NO--INSERT SYMBOL IN USER SYMBOL TABLE
0075 217 MOVZBL #ID,R8 ;RESULT ALWAYS ID
0075 218 10$: RSB
0075 219
0075 220 .SBTTL MAC$SYMNUM LETTERS A-F CAN START SYMBOL OR NUMBER
0075 221
0075 222 MAC$SYMNUM::
0075 223 CMPB W^MAC$GB_RDXNDX,- ;ARE WE GOING TO GET HEX NUMBER?
0075 224 #RDXSV_HEX
0075 225 BEQL MAC$NUMBER ;IF EQL YES
0075 226 CMPB R10,#^A/B/ ;CAN SYMBOL BE 'B^'?
0075 227 BEQL MAC$XSYMBL ;YES--SEE IF 'B^'
0075 228 CMPB R10,#^A/b/ ;OR LOWER CASE 'b^'?
0075 229 BEQL MAC$XSYMBL ;YES--SEE IF 'b^'
0075 230 BRW MAC$SYMBOL ;NO--GO SCAN SYMBOL
0075 231
0075 232 .SBTTL MAC$XSYMBL NEXT CHARACTER CAN BE AN UPARROW
0075 233
0075 234 :++
0075 235 : CHECK IF NEXT CHARACTER IS AN UPARROW. IF IT IS THEN THIS IS
0075 236 : A WIDTH INDICATOR.
0075 237 :
0075 238 :--
0075 239
0075 240 MAC$XSYMBL::
0075 241 CMPB @W^MAC$GL_LINEPT,#^A/^/ ;NEXT CHAR AN UPARROW?
0075 242 BEQL 20$ ;IF EQL YES--CONTINUE
0075 243 10$: BRW MAC$SYMBOL ;NO--GO SCAN A SYMBOL
0075 244 20$: LOCC R10,#LEN$K_XUPTAB,- ;LOOKUP CHARACTER IN XUPTAB
0075 245 G^MAC$AB_XUPTAB ; to see if it can preceded '^'
0075 246 BEQL 10$ ;IF EQL NOT IN TABLE--SCAN SYMBOL
0075 247 SUBL3 #MAC$AB_XUPTAB,R1,R2 ;GET INDEX INTO TABLE FOR CHARACTER
0075 248 MOVZBL MAC$AB_XUPTOKEN[R2],R8 ;RETRIEVE TOKEN VALUE
0075 249 BSBB 30$ ;SCAN OVER THE CHARACTER
0075 250 30$: BRW MAC$GETCHR ;SCAN OVER UPARROW AND RETURN
0075 251 ; WITH TOKEN VALUE
0075 252
0075 253 .SBTTL MAC$NUMBER ACCUMULATE NUMBER
0075 254
```

02DC 30
58 D5
06 12
01D1 30
58 0C 9A
05

0000'CF 91
03
38 13
42 8F 5A 91
09 13
62 8F 5A 91
03 13
FF67 31

5E 8F 0000'DF 91
03 13
FF5C 31
0000'8F 5A 3A
00000000'GF 00A9
F1 13
52 51 00000000'8F C3
58 00000000'EF42 9A
00 10
FF3B' 31

```
50      0000'CF  9A 00C5 255 MAC$NUMBER::
50      00D2'CF40 D0 00CA 256      MOVZBL W*MAC$GB RDXNDX,R0      ;GET INDEX FOR NUMBER GETTER
        60      17 00D0 257      MOVL   W*NUM_TAB[R0],R0      ;GET ADDRESS OF ROUTINE TO READ NUMBER
        00D2 258      JMP      (R0)      ;AND GO GET THE NUMBER
        000004FC' 00D2 259
        00000504' 00D6 260 NUM_TAB:.LONG MAC$BINNUM      ;(0) BINARY
        00000421' 00DA 261      .LONG MAC$OCTNUM      ;(1) OCTAL
        0000050C' 00DE 262      .LONG MAC$DNUMBER      ;(2) DECIMAL
        00000000' 00E2 263      .LONG MAC$HEXNUM      ;(3) HEX
        00000000' 00E6 264      .LONG MAC$GETFLOAT      ;(4) F FLOATING
        00000000' 00EA 265      .LONG MAC$GETDOUBLE      ;(5) D FLOATING
        00000000' 00EE 266      .LONG MAC$GETGFLOAT      ;(6) G FLOATING
        00000000' 00EE 267      .LONG MAC$GETHFLOAT      ;(7) H FLOATING
```

```
00F2 269 .SBTTL MAC$SYMSCN SCAN FOR SYMBOL NAME
00F2 270
00F2 271 :++
00F2 272 THIS ROUTINE IS CALLED BY SEVERAL DIRECTIVES TO SCAN
00F2 273 A NAME FROM THE INPUT LINE. LEADING SPACES AND TABS
00F2 274 ARE IGNORED. IF NOT FOUND, R0 IS SET TO 0.
00F2 275 :--
00F2 276
00F2 277
00F2 278 .ENABL LSB
00F2 279 MAC$SYMSCNUP::
11 18 10 00F2 280 BSBB MAC$SYMSCN ;SCAN SYMBOL NAME
50 50 E8 00F4 281 BLBS R0,5$ ;IF LBS CVT TO UPPER CASE
24 11 00F7 282 BRB 30$ ;ELSE JUST RETURN
00F9 283
00F9 284 MAC$SYMSCNUP1::
11 10 00F9 285 BSBB MAC$SYMSCN ;SCAN THE SYMBOL NAME
55 DD 00FB 286 PUSHL R5 ;SAVE R5 OVER MOVC
0000'CF 20 28 00FD 287 MOVC3 #SYMSK_MAXLEN+1,W*MAC$AB_TMSYM,- ;COPY NAME TO TEMP BUFFER
0000'CF 55 BED0 0102 288 W*MAC$AB_TMSY1
35 10 0108 289 POPL R5 ;RESTORE R5
OE 11 010A 290 5$: BSBB MAC$CVT_SYM_UP ;CONVERT TO UPPER CASE
010C 291 BRB 20$ ;AND RETURN
010C 292
010C 293 MAC$SYMSCN::
FEF1' 30 010C 294 BSBW MAC$SKIPSP ;SKIP LEADING SPACES
02 E0 010F 295 BBS #CHRSV_SYM CHR,- ;BRANCH IF CHAR CAN START A SYMBOL
03 0000'CA 0111 296 W*MAC$AB_CMSK_TAB(R10),10$
50 D4 0115 297 CLRL R0 ;NO--RETURN NO SYMBOL SCANNED
05 0117 298 RSB
57 10 0118 299 10$: BSBB MAC$GETSYM ;YES--SCAN THE SYMBOL
50 01 9A 011A 300 20$: MOVZBL #1,R0 ;RETURN SYMBOL SCANNED
05 011D 301 30$: RSB
011E 302 .DSABL LSB
011E 303
011E 304 .SBTTL MAC$LCLSKIP SCAN FOR LOCAL LABEL
011E 305
011E 306 :++
011E 307 This routine skips over a string which looks like a local label.
011E 308 A false status return indicates that a syntax error was detected.
011E 309 :--
011E 310 .ENABL LSB
011E 311
011E 312 MAC$LCLSKIP::
FEDF' 30 011E 313 bsbw MAC$SKIPSP ;Skip leading spaces
04 E1 0121 314 bbc #CHRSV_NUM BER,- ;Local labels start with a decimal no.
15 0000'CA 0123 315 W*MAC$AB_CMSK_TAB(R10),20$
FED6' 30 0127 316 10$: bsbw MAC$GETCHR ;Move to next character
04 E0 012A 317 bbs #CHRSV_NUM BER,- ;Continue to consume decimal digits.
F7 0000'CA 012C 318 W*MAC$AB_CMSK_TAB(R10),10$
24 5A 91 0130 319 cmpb r10,#a/$/ ;Ensure trailing Dollar Sign.
07 12 0133 320 bneq 20$
FEC8' 30 0135 321 bsbw MAC$GETCHR ;Consume it.
50 01 9A 0138 322 movzbl #1,r0 ;The syntax was correct.
05 013B 323 rsb
50 D4 013C 324 20$: clrl r0 ;Local Label was not found.
05 013E 325 rsb
```



```
013F 326
013F 327 .DSABL LSB
013F 328
013F 329 .SBTTL CONVERT SYMBOL TO UPPER CASE FOR TABLE SEARCH
013F 330
013F 331 :++
013F 332 :FUNCTIONAL DESCRIPTION:
013F 333 :
013F 334 :THIS ROUTINE CONVERTS THE SYMBOL IN MAC$AB_TMSYM
013F 335 :TO ALL UPPER CASE CHARACTERS AND COMPUTES A NEW HASH VALUE
013F 336 :FOR THE SYMBOL.
013F 337 :
013F 338 :--
013F 339
013F 340 MAC$CVT_SYM_UP::
51 0000'CF 9E 013F 341 MOVAB W*MAC$AB_TMSYM,R1 ;POINT TO THE SYMBOL
52 81 9A 0144 342 MOVZBL (R1)+,R2 ;GET SYMBOL CHARACTER COUNT
50 52 D0 0147 343 MOVL R2,R0 ;INITIALIZE HASH VALUE TO COUNT
53 81 9A 014A 344 10$: MOVZBL (R1)+,R3 ;GET NEXT CHARACTER OF NAME
61 8F 53 91 014D 345 CMPB R3,#^A/A/ +^X20 ;CAN CHARACTER BE LOWER CASE?
OD 1F 0151 346 BLSSU 20$ ;IF LSSU NO
7A 8F 53 91 0153 347 CMPB R3,#^A/Z/ +^X20 ;STILL CHECKING FOR LOWER CASE
07 1A 0157 348 BGTRU 20$ ;IF GTRU NOT LOWER CASE
53 20 8A 0159 349 BICB2 #^X20,R3 ;YES--CONVERT TO UPPER CASE
FF A1 53 90 015C 350 MOVB R3,-1(R1) ;STORE IMPROVED CHARACTER
50 53 C0 0160 351 20$: ADDL2 R3,R0 ;IMPROVE HASH VALUE
E4 52 F5 0163 352 SOBGTR R2,10$ ;LOOP FOR WHOLE NAME
0000'CF 50 FFFFFFFF80 8F CB 0166 353 BICL3 #^C<HASHSZ>,R0,W*MAC$GL_HSHVAL ;TRIM TO HASH TABLE SIZE
05 0170 354 RSB
```

```
0171 356 .SBTTL MAC$GETSYM ACCUMULATE SYMBOL NAME
0171 357
0171 358 **
0171 359 FUNCTIONAL DESCRIPTION:
0171 360
0171 361 THIS ROUTINE READS THE INPUT STREAM AND ACCUMULATES A SYMBOL
0171 362 NAME AND ITS HASH VALUE.
0171 363
0171 364 CALLING SEQUENCE:
0171 365
0171 366 JSB MAC$GETSYM
0171 367
0171 368 INPUT PARAMETERS:
0171 369
0171 370 R10 FIRST CHARACTER OF SYMBOL NAME
0171 371
0171 372 IMPLICIT INPUTS:
0171 373
0171 374 NONE
0171 375
0171 376 OUTPUT PARAMETERS:
0171 377
0171 378 MAC$AB_TMSYM CONTAINS: .BYTE SYMLEN,.ASCII/SYMBOLNAME/
0171 379 MAC$GL_HSHVAL THE HASH VALUE FOR THE ACCUMULATED SYMBOL
0171 380
0171 381 IMPLICIT OUTPUTS:
0171 382
0171 383 NONE
0171 384
0171 385 COMPLETION CODES:
0171 386
0171 387 NONE
0171 388
0171 389 SIDE EFFECTS:
0171 390
0171 391 NONE
0171 392
0171 393 --
0171 394
0171 395
0171 396 MAC$GETSYM::
0171 397
56 0070 8F BB 0171 397 PUSHR #*M<R4,R5,R6> ;SAVE REGISTERS
00 0000 CF 9E 0175 398 MOVAB W*MAC$AB_TMSYM,R6 ;POINT TO TEMP SYMBOL STORAGE
6E 00 2C 017A 399 MOVCS #0,(SP),#0,- ;CLEAR THE TEMPORARY SYMBOL NAME
66 20 017E 400 #SYM$K_MAXLEN+1,(R6)
55 01 A6 9E 0180 401 MOVAB 1(R6),R5 ;POINT TO FIRST CHAR OF NAME
0184 402 ; (FIRST BYTE HOLDS SYMBOL LENGTH)
1E DD 0184 403 PUSHL #SYM$K_MAXLEN-1 ;SET MAX LENGTH (1ST CHAR COUNTED)
85 5A 90 0186 404 MOVB R10,(R5)+ ;STORE FIRST CHAR OF NAME
54 5A 9A 0189 405 MOVZBL R10,R4 ;GET FIRST CHAR AND INIT HASH VALUE
018C 406
018C 407 ; LOOP, COLLECTING SYMBOL NAME AND HASH VALUE
018C 408
FE71' 30 018C 409 10$: BSBW MAC$GETCHR ;GET NEXT CHARACTER OF NAME
03 E1 018F 410 BBC #CHRSV_SYM_CH1,- ;BRANCH IF NOT LEGAL FOR SYMBOL
OC 0000 CA 0191 411 W*MAC$AB_CMSK_fAB(R10),30$
6E D7 0195 412 20$: DECL (SP) ;COUNT THE CHARACTER
```

```
      F3  19  0197  413      BLSS  10$      ;IF LSS NAME IS TOO LONG--DO NOT
      85  5A  90  0199  414      ;STORE
      54  5A  C0  0199  415      MOVB  R10,(R5)+ ;STORE BYTE IN TEMP SYM BLOCK
      EB  11  019C  416      ADDL2 R10,R4      ;ACCUM NEW HASH VALUE
      019F  417      BRB  10$      ;LOOP FOR END OF SYMBOL
      01A1  418      ;
      01A1  419      ; NAME IS ACCUMULATED
      01A1  420      ;
      53  8E  D0  01A1  421 30$: MOVL  (SP)+,R3      ;GET <15-LENGTH>
      18  18  01A4  422      BGEQ  40$      ;IF GEQ LEGAL SYMBOL LENGTH
      53  D4  01A6  423      CLRL  R3      ;TOO LONG--MAKE IT MAX LENGTH
      0000'CF D5  01A8  424      TSTL  W^MAC$GL_MCLVL ;ARE WE IN A MACRO DEFINITION?
      10  14  01AC  425      BGTR  40$      ;IF GTR YES--SKIP ERROR MSG
      01AE  426      $INTOUT LW INT$ WRN,- ;ISSUE WARNING IN PASS 2
      01AE  427      <#MAC$_TLLSYMLEN,W^MAC$GL_LINEPT> ; Symbol name exceeds max length
      53  1F  53  C3  01BE  428 40$: SUBL3 -R3,#SYMSK_MAXLEN,R3 ; COMPUTE # CHARACTERS IN SYMBOL
      66  53  90  01C2  429      MOVB  R3,(R6) ;STORE # OF CHARS IN MAC$AB_TMP$SYM
      54  53  C0  01C5  430      ADDL2 R3,R4      ;HASH VALUE IS SUM OF CHARACTERS
      0000'CF 54  FFFFFFF80 8F  CB  01C8  431      BICL3 #^C<HASHSZ>,R4,- ;PLUS THE NUMBER OF CHARACTERS
      0070 8F  BA  01D2  432      POPR  W^MAC$GL_HSHVAL ;TRIM TO HASH TABLE SIZE
      05  01D6  434      RSB  #^M<R4,R5,R6> ;AND STORE IN HASH VALUE
      01D6  435
```



```
01D7 437 .SBTTL MAC$XUPARROW CHECK CHARACTER AFTER '^'
01D7 438
01D7 439 :++
01D7 440 THIS ROUTINE RETURNS IN R8 THE TOKEN CORRESPONDING TO THE
01D7 441 CHARACTER FOLLOWING THE UPARROW. IF THE CHARACTER IS NOT
01D7 442 IN THE TABLE THIS ROUTINE BRANCHES TO MAC$CHRRER TO REPORT
01D7 443 THE ILLEGAL CHARACTER.
01D7 444
01D7 445 NOTE THAT THIS ROUTINE IS JMP'ED TO BY MAC$TOKEN.
01D7 446 :--
01D7 447
01D7 448 MAC$XUPARROW::
01D7 449 BSBW MAC$GETCHR ;GET THE NEXT CHARACTER
01DA 450 LOCC R10,#LENSK UPXTAB,- ;LOOK UP CHARACTER IN UPXTAB
01DF 451 G*MAC$AB_UPXTAB ; to see if legal character
01E4 452 BEQL 10$ ;IF EQL THEN ILLEGAL CHARACTER
01E6 453 SUBL3 #MAC$AB_UPXTAB,R1,R2 ;GET INDEX INTO TABLE FOR CHARACTER
01EE 454 MOVZBL MAC$AB_UPXTOKEN[R2],R8 ;RETRIEVE TOKEN VALUE
01F6 455 BRW MAC$GETCHR ;GET THE FIRST REAL CHARACTER
01F9 456 ;AND RETURN
01F9 457
01F9 458 : CHARACTER IS ILLEGAL
01F9 459 10$: BRW MAC$CHRRER ;REPORT ILLEGAL CHARACTER
01FC 461
01FC 462 .SBTTL MAC$XPOUND A POUND SIGN WAS DETECTED
01FC 463
01FC 464 :++
01FC 465 THIS ROUTINE LOADS MAC$GB_RDXNDX BASED ON THE OPCODE TYPE
01FC 466 FOUND IN MAC$GL_MOPPTR.
01FC 467 :--
01FC 468
01FC 469 MAC$XPOUND::
01FC 470 MOVL W*MAC$GL_MOPPTR,R0 ;GET PTR TO OPERAND DESCR. BYTE
0201 471 BEQL 10$ ;IF EQL NOT FLOATING POINT
0203 472 TSTW (R0) ;FLOATING OPERAND?
0205 473 BGEQ 10$ ;IF GEQ NO
0207 474 BBC #OPDSV_H_FLOAT,(R0),2$ ;IF NOT H FLOAT THEN SKIP
020B 475 MOVB #RDXSV_HFLOAT,- ;YES--USE H FLOAT READER
020D 476 W*MAC$GB_RDXNDX
0210 477 BRB 20$
0212 478 BBC #OPDSV_G_FLOAT,(R0),4$ ;IF NOT G FLOAT THEN SKIP
0216 479 MOVB #RDXSV_GFLOAT,- ;YES--USE G FLOAT READER
0218 480 W*MAC$GB_RDXNDX
021B 481 BRB 20$
021D 482 BBC #OPDSV_D_FLOAT,(R0),6$ ;IF NOT D FLOAT THEN SKIP
0221 483 MOVB #RDXSV_DOUBLE,- ;YES--USE D FLOAT READER
0223 484 W*MAC$GB_RDXNDX
0226 485 BRB 20$
0228 486 MOVB #RDXSV_FLOAT,- ;USE F FLOAT READER
022A 487 W*MAC$GB_RDXNDX
022D 488 BRB 20$
022F 489 10$: MOVB #RDXSV_DECIMAL,- ;ELSE USE DECIMAL
0231 490 W*MAC$GB_RDXNDX
0234 491 20$: MOVZBL #DPOUND,R8 ;RETURN TOKEN OF POUND SIGN
0237 492 BRW MAC$GETCHR ;READ NEXT CHARACTER AND RETURN
```

```
023A 494 .SBTTL INSERT/SEARCH USER SYMBOL TABLE
023A 495
023A 496
023A 497 :++
023A 498 FUNCTIONAL DESCRIPTION:
023A 499
023A 500 THIS ROUTINE IS CALLED TO LOOKUP A SYMBOL IN THE USER SYMBOL
023A 501 TABLE, AND IF THE ENTRY IS AT MAC$INSUSRSYMTB, THEN TO INSERT
023A 502 THE SYMBOL IF IT IS NOT FOUND.
023A 503
023A 504 CALLING SEQUENCE:
023A 505
023A 506 OR JSB MAC$INSUSRSYMTB
023A 507 JSB MAC$SRCUSRSYMTB
023A 508
023A 509 INPUT PARAMETERS:
023A 510
023A 511 MAC$AB_TPSYM CONTAINS: .BYTE SYMLEN,.ASCII/SYMBOLNAME/
023A 512 MAC$GL_HSHVAL THE HASH VALUE FOR THE ACCUMULATED SYMBOL
023A 513 R3 HASH TABLE ADDRESS IF ENTRY AT MAC$SRCSYMTAB
023A 514 R5 POINTER TO LINKED LIST IF ENTRY AT MAC$SRC LIST
023A 515 R6 POINTER TO WHERE R5 POINTS IF ENTRY AT MAC$SRC LIST
023A 516 AND YOU NEED THE PREV. POINTER RETURNED
023A 517
023A 518 OUTPUT PARAMETERS:
023A 519
023A 520 R0 0 NOT FOUND
023A 521 1 FOUND
023A 522 R1 SYMBOL BLOCK ADDRESS IF FOUND
023A 523 0 IF NOT FOUND
023A 524 R2 POINTER TO PREVIOUS SYMBOL BLOCK
023A 525 :--
023A 526
023A 527 .ENABL LSB
023A 528 MAC$SRC_LIST:: :ENTRY TO SEARCH LINKED LIST
023A 529 BICL2 #FLG$M_ORDLST!FLG$M_INSERT,- :NOT ORDERED LIST
0240 530 (R11) :AND DO NOT INSERT
0241 531 PUSHR #*M<R4,R5,R6,R7,R8> :SAVE REGISTERS
0245 532 BRB 30$ :JUMP INTO COMMON CODE
0247 533 MAC$SRCSYMTAB:: :ENTRY TO SEARCH ARBITRARY TABLE
0247 534 BICL2 #FLG$M_ORDLST!FLG$M_INSERT,- :OPCODE TABLE IS NOT ORDERED
024D 535 (R11) :AND DO NOT ALLOW INSERTION
024E 536 BRB 20$
0250 537
0250 538 MAC$INSUSRSYMTB:: :ENTRY TO INSERT IN USER SYMBOL TABLE
0250 539 BBBS #FLG$V_INSERT,(R11),10$ :SET INSERT FLAG
0254 540 BRB 10$ :JOIN COMMON CODE
0256 541
0256 542 MAC$SRCUSRSYMTB:: :ENTRY TO SEARCH USER SYMBOL TABLE
0256 543 BBCC #FLG$V_INSERT,(R11),10$ :DO NOT INSERT IF NOT FOUND
025A 544 10$: MOVAB W*MAC$AL_USYH$HTB,R3 :PICK UP SYMBOL HASH TABLE ADDRESS
025F 545 BBBS #FLG$V_ORDLST,(R11),20$ :USER SYMBOL TABLE IS ORDERED
0263 546 20$: PUSHR #*M<R4,R5,R6,R7,R8> :SAVE REGISTERS
0267 547 MOVL W*MAC$GL_HSHVAL,R4 :GET HASH VALUE FOR SYMBOL
026C 548 MOVAL (R3)[R4],R6 :GET ADDRESS OF PREVIOUS SYMBOL
0270 549 MOVL (R6),R5 :POINT TO FIRST SYMBOL
0273 550 BEQL 70$ :IF EQL NOTHING IN BUCKET
```

00020100 8F CA 023A 528
01F0 8F BB 0241
2E 11 0245
00020100 8F CA 0247
68 024D
13 11 024E
0250
06 6B 08 E3 0250
04 11 0254
0256
00 6B 08 E5 0256
53 0000'CF 9E 025A
00 6B 11 E3 025F
01F0 8F BB 0263
54 0000'CF D0 0267
56 6344 DE 026C
55 66 D0 0270
2B 13 0273

```
57 0000'CF 9E 0275 551 30$: MOVAB W*MAC$AB_TMP$SYM,R7 ;POINT TO TEMP SYMBOL NAME
58 87 9A 027A 552 MOVZBL (R7)+,R8 ;GET # CHARS IN NEW SYMBOL NAME
027D 553
027D 554 : CHAIN THROUGH THE BUCKET LOOKING FOR THE SYMBOL. GET OUT IF WE FIND
027D 555 : IT OR IF WE FIND A SYMBOL LARGER THAN THIS ONE, SINCE THE BUCKETS
027D 556 : ARE LINKED FROM SMALLEST TO LARGEST.
027D 557
50 04 A5 9A 027D 558 40$: MOVZBL SYMB$NAME(R5),R0 ; Get offset to symbol count/name
50 55 50 C3 0281 559 SUBL3 R0,R5,R0 ; and form its address
51 80 9A 0285 560 MOVZBL (R0)+,R1 ; Get symbol size and advance pointer
60 51 00 67 58 2D 0288 561 CMPC5 R8,(R7),#0,R1,(R0) ; Match the symbols
17 13 028E 562 BEQL 75$ ; Branch if symbol found
0A 1F 0290 563 50$: BLSSU 65$ ; IF SYMBOL LESS THE ONE IN THE TABLE
THEN GET OUT.
56 55 D0 0292 564 60$: MOVL R5,R6 ;BRING UP THE BACK POINTER
55 65 D0 0295 566 MOVL SYMB$_LINK(R5),R5 ; Link to next symbol block
E3 12 0298 567 BNEQ 40$ ; IF NEQ GO GET IT
04 11 029A 568 BRB 70$ ; Symbol not found
029C 569
F2 6B 11 E1 029C 570 65$: BBC #FLG$V_ORDLST,(R11),60$ ;BR IF NOT ORDERED LIST
02A0 571
02A0 572 : THE SYMBOL IN QUESTION IS NOT IN THE BUCKET
02A0 573
06 6B 08 E4 02A0 574 70$: BBSC #FLG$V_INSERT,(R11),80$ ;SYMBOL NOT THERE--SHOULD WE INSERT?
00A3 31 02A4 575 BRW 160$ ;NO--GO RETURN NOT FOUND
0092 31 02A7 576 75$: BRW 120$ ;FOUND IT--GO FINISH UP
57 0000'CF D0 02AA 577 80$: MOVL W*MAC$GL_SYMPGPTR,R7 ;YES--GET POINTER TO CURRENT PAGES
41 12 02AF 578 BNEQ 100$ ;IF NEQ GO USE THEM
02B1 579
02B1 580 : ALLOCATE PAGES FOR THIS BUCKET
02B1 581
00001400 8F DD 02B1 582 90$: PUSHL #<512*STB$K_PG_MISS> ;STACK SIZE OF BLOCK WE NEED
52 5E D0 02B7 583 MOVL SP,R2 ;SAVE ADDRESS
0000'CF 9F 02BA 584 PUSHAB W*MAC$GL_BASEADDR ;STACK RESULT LOCATION
52 DD 02BE 585 PUSHL R2 ;STACK ADDRESS OF SIZE
00000000'GF 02 FB 02C0 586 CALLS #2,G*LIB$GET_VM ;ALLOCATE BLOCK
8E D5 02C7 587 TSTL (SP)+ ;KEEP STACK CLEAN
7B 50 E9 02C9 588 BLBC R0,150$ ;IF LBC ALLOCATION FAILURE
50 0000'CF D0 02CC 589 MOVL W*MAC$GL_SYMPGPTR,R0 ;GET LAST PAGES ALLOCATED
05 13 02D1 590 BEQL 92$ ;IF EQL THIS IS FIRST ALLOCATE
0000'CF 60 0E 02D3 591 INSQUE (R0),W*MAC$GL_SYM_PAGL ;LINK INTO SYMBOL PAGE LIST
0000'CF D6 02D8 592 92$: INCL W*MAC$GL_SYMPGREQ ;COUNT ANOTHER SYMBOL PAGES GET
50 0000'CF D0 02DC 593 MOVL W*MAC$GL_BASEADDR,R0 ;GET ADDRESS OF BLOCK ALLOCATED
57 50 D0 02E1 594 MOVL R0,R7 ;COPY ADDRESS FOR LATER
0000'CF 50 D0 02E4 595 MOVL R0,W*MAC$GL_SYMPGPTR ;STORE POINTER TO CURRENT PAGES
80 13FB 8F 3C 02E9 596 MOVZWL #<<512*STB$K_PG_MISS>-8>,(R0)+ ; Set count of bytes in block
02EE 597 : IN THESE PAGES.
80 04 A0 9E 02EE 598 MOVAB 4(R0),(R0)+ ;STORE POINTER TO FIRST SYMBOL BLOCK
02F2 599
02F2 600 : SYMBOL PAGE IS POINTED TO BY R7
02F2 601
02F2 602
50 0000'CF 9E 02F2 603 100$: MOVAB W*MAC$AB_TMP$SYM,R0 ; Get address of new symbol count/name
51 60 9A 02F7 604 MOVZBL (R0),R1 ; Get symbol count and advance pointer
10 51 91 02FA 605 CMPB R1,#SYMB$_TWO$COL ; Will 2 column symbol table be required?
04 19 02FD 606 BLSS 110$ ; No if LSS
00 6B 2A E2 02FF 607 BBSS #FLG$V_SYM2COL,(R11),110$ ; Set bit for 2 column symbol table
```



```

      0303 608 110$:
67 51 D6 0303 609 INCL R1 ; Include count byte in length
87 0D C2 0305 610 SUBL2 R1,(R7) ; Subtract length and
      A4 19 0308 611 SUBL2 #SYMSK_BLKSIZE,(R7)+ ; and fixed part from bytes in block
58 67 D0 030B 612 BLSS 90$ ; If LSS no space left so get new block
67 51 C0 030D 613 MOVL (R7),R8 ; Get new block address
67 0D C0 0310 614 ADDL2 R1,(R7) ; Add size of this block to figure
7E 51 90 0313 615 ADDL2 #SYMSK_BLKSIZE,(R7) ; address of next
68 60 51 28 0316 616 MOVB R1,-(SP) ; Save total length
58 53 D0 0319 617 MOVC3 R1,(R0),(R8) ; Copy symbol count/name into block
83 66 D0 031D 618 MOVL R3,R8 ; Set proper block address
66 58 D0 0320 619 MOVL (R6),(R3)+ ; Link new symbol block
83 8E 90 0323 620 MOVL R8,(R6) ; into bucket
      83 94 0326 621 MOVB (SP)+,(R3)+ ; Store name offset
83 0000'CF D0 0329 622 MOVL W*MAC$GL_VALUE,(R3)+ ; Put symbol value in
      83 B4 032E 623 CLRW (R3)+ ; Clear symbol's flags
      83 94 0330 624 CLRB (R3)+ ; Token value
83 0000'CF 90 0332 625 MOVB W*MAC$GL_PSECT,(R3)+ ; Segment number
      51 58 D0 0337 626 ;
      03 11 033A 627 MOVL R8,R1 ; Point to symbol block
      033C 628 BRB 140$ ;EXIT ROUTINE
      033C 629 ;
      033C 630 ; SYMBOL WAS FOUND IN THE TABLE
      033C 631 ;
51 55 D0 033C 632 120$: MOVL R5,R1 ;POINT TO SYMBOL BLOCK
50 01 9A 033F 633 140$: MOVZBL #1,R0 ;RETURN SUCCESS
52 56 D0 0342 634 MOVL R6,R2 ;POINT TO PREVIOUS SYMBOL
      08 11 0345 635 BRB 170$ ;EXIT ROUTINE
      0347 636 ;
      0347 637 ; PAGE ALLOCATION FAILURE
      0347 638 ;
      FCB6' 31 0347 639 150$: BRW MAC$ERR_NOMEM ;NO MEMORY--GO REPORT ERROR
      034A 640 ;
      034A 641 ; SYMBOL NOT FOUND AND NOT INSERTING
      034A 642 ;
52 50 7C 034A 643 160$: CLRQ R0 ;RETURN 0 ON FAILURE
01F0 56 D0 034C 644 MOVL R6,R2 ;RETURN POINTER TO PREVIOUS
      8F BA 034F 645 170$: POPR #*M<R4,R5,R6,R7,R8> ;RESTORE REGISTERS
      05 0353 646 RSB
      0354 647
      0354 648 .DSABL LSB
```

```
.SBTTL MAC$REGIS SEE IF THE SCANNED SYMBOL IS A REGISTER

THIS ROUTINE CHECKS THE SYMBOL NAME TO SEE IF IT IS A REGISTER.
R8 WILL BE 0 IF IT IS NOT, AND NON-ZERO IF IT IS. R1 WILL
CONTAIN THE VALUE.

MAC$REGIS:
55 0000'CF 9E 0354 650
02 01 85 8F 0354 651
0009' 0354 652
0012' 0354 653
003A' 0354 654
0354 655
0354 656
0354 657
0354 658
0354 659
0354 660
0359 661
035D 662
035F 663
0361 664
0363 665
0363 666
0363 667
0363 668
0365 669
0366 670
0366 671
0366 672
0366 673
0369 674
036B 675
036E 676
036F 677
036F 678
036F 679
036F 680
0373 681
0376 682
0378 683
0378 684
0378 685
0378 686
037E 687
0382 688
0389 689
038C 690
0391 691
0393 692
0396 693
0397 694
0397 695
0397 696
0397 697
039C 698
039E 699
03A2 700
03A4 701
03A7 702
03A9 703
03AC 704
03AF 705
03B2 706

MAC$REGIS:
MOVAB W^MAC$AB_TMP$SYM,R5 ;POINT TO SYMBOL LENGTH+NAME
CASEB (R5)+,#1,#2 ;SEE IF 1-3 CHARACTER NAME
10$: .WORD 30$-10$ ;ONE-CHARACTER NAME
.WORD 40$-10$ ;TWO-CHARACTER NAME
.WORD 70$-10$ ;THREE-CHARACTER NAME

DEFINITELY NOT A REGISTER--RETURN 0
20$: CLRL R8 ;RETURN 0 FOR NOT A REGISTER
RSB

ONE CHARACTER NAME--IS IT '.'?
30$: CMPB (R5),#^A/. / ;IS SYMBOL CURRENT PC (.)?
BNEQ 20$ ;IF NEQ NO
MOVZBL #DPC,R8 ;YES--RETURN TOKEN FOR PC
RSB

TWO CHARACTER NAME CAN BE R0-R9, SP, PC, AP, FP, IV, OR DV
40$: MATCHC #2,(R5),#TWO CHR REG SZ,- ;SEE IF A REGISTER
W^TWO_CHR_REG_NAM ;IF NEQ NOT IN TABLE
BNEQ 20$ ;IF LBS THEN NAME IS LAST CHAR
BLBS R2,20$ ;OF ONE REG AND FIRST OF ANOTHER
(EX. PP, VD, CA)
DIVL2 #2,R2 ;FIGURE BYTE INDEX
SUBL3 R2,#15,R0 ;FIGURE INDEX INTO TWO CHR REG_VAL TABLE
MOVZBL L^TWO_CHR_REG_VAL(R0),R1 ;PICK UP THE REGISTER VALUE
50$: MOVZBL #RRREG,R8 ;RETURN TOKEN OF REGISTER
CMPB 1(R5),#^A/V/ ;UNLESS 'IV' OR 'DV'
BNEQ REG CREF ;IF NEQ NOT 'IV' OR 'DV'--SEE IF CREF
MOVZBL #DMASK,R8 ;'IV' OR 'DV'--TOKEN IS DMASK
RSB
60$:

THREE CHARACTER SYMBOL NAME CAN BE R10-R12
70$: CMPW (R5)+,#^A/R1/ ;IS IT A REGISTER?
BNEQ 20$ ;IF NEQ NO
SUBB3 #^A/O/,(R5),R1 ;YES--GET VALUE IN R1
BLSS 20$ ;IF LSS THEN NOT REGISTER
CMPB R1,#2 ;IN RANGE?
BGTR 20$ ;IF GTR NO
ADDB2 #10.,R1 ;YES--ADJUST REGISTER VALUE
MOVZBL #RRREG,R8 ;RETURN TOKEN IS REGISTERS
MOVZBL R1,R1 ;EXTEND REG. NUMBER TO LONGWORD
BRB REG_CREF ;SEE ABOUT CREF AND RETURN
```

```
03B2 708 :++
03B2 709 : FUNCTIONAL DESCRIPTION:
03B2 710 :
03B2 711 :     THIS ROUTINE CROSS-REFERENCES REGISTERS IF REGISTER CROSS-REF
03B2 712 :     IS ENABLED.
03B2 713 :
03B2 714 : INPUTS:
03B2 715 :
03B2 716 :     R1     REGISTER NUMBER
03B2 717 :
03B2 718 : --
03B2 719 :
03B2 720 REG_CREF:
03B2 721 BBC      #CRFSV_REGISTERS,W*MAC$GL_CRF_FLG,30$ ;BRANCH IF NOT CREFFING REGS
03B2 722 BBS      #FLGSV_XCRF,(R1),30$ ;BRANCH IF .NOCROSS ON
03B2 723 PUSHL    R1 ;CREFFING--SAVE R1
03B2 724 MOVL     L*MAC$AL_RGNM_TAB[R1],R5 ;POINT R5 TO REGISTER SYMBOL BLOCK
03B2 725 INCL     SYMSL_VAL(R5) ;COUNT REGISTER REFERENCES
03B2 726 BBS      #SYMSV_CRFO,SYMSW_FLAG(R5),10$ ;BRANCH IF INSERT DONE
03B2 727 CLRL     -(SP) ;NO--CLEAR FLAGS
03B2 728 PUSHAB   SYMSL_VAL(R5) ;POINT TO VALUE WORD
03B2 729 MOVZBL   SYMSB_NAME(R5),R0 ; Get offset to symbol name/count
03B2 730 SUBL3     R0,R5,-(SP) ; and form its address on stack
03B2 731 PUSHAB   W*MAC$AL_CRFRCGTB ;STACK CONTROL TABLE ADDRESS
03B2 732 CALLS    #4,G*CRF$INSRTKEY ;INSERT REGISTER KEY
03B2 733 10$:   PUSHL    #CRFSK_REF ;THIS IS A REFERENCE
03B2 734       PUSHL    #0 ;ASSUME READ REFERENCE
03B2 735 MOVL     W*MAC$GL_MOPPTR,R0 ;GET OPERAND MODE BYTE POINTER
03B2 736 BEQL     20$ ;IF EQL NOT IN INSTRUCTION
03B2 737 BITW     #OPDSM_MODIFY!OPDSM_WRITE,(R0) ;MODIFY OR WRITE REF?
03B2 738 BEQL     20$ ;IF EQL NO
03B2 739 MOVL     #1,(SP) ;YES--SET WRITE REFERENCE
03B2 740 20$:   PUSHAB   W*MAC$AB_LPBUF ;REFERENCER NAME ADDRESS
03B2 741 MOVZBL   SYMSB_NAME(R5),R0 ; Get offset to symbol name/count
03B2 742 SUBL3     R0,R5,-(SP) ; and form its address on stack
03B2 743 PUSHAB   W*MAC$AL_CRFRCGTB ;CREF CONTROL TABLE ADDRESS
03B2 744 BSBW     MAC$CVT [IN PAG ;CONVERT LINE/PAGE INTO LPBUF
03B2 745 CALLS    #5,G*CRF$INSRTREF ;INSERT REGISTER REFERENCE
03B2 746 INCL     W*MAC$GL_CRF_RCNT ;COUNT REGISTER REFERENCE
03B2 747 POPL     R1 ;RESTORE REGISTER NUMBER
03B2 748 30$:   RSB
```

68 0000'CF 03 E1 03B2 721 BBC #CRFSV_REGISTERS,W*MAC\$GL_CRF_FLG,30\$;BRANCH IF NOT CREFFING REGS
64 6B 1F E0 03B8 722 BBS #FLGSV_XCRF,(R1),30\$;BRANCH IF .NOCROSS ON
55 00000030'EF41 D0 03B2 723 PUSHL R1 ;CREFFING--SAVE R1
05 A5 D6 03BE 724 MOVL L*MAC\$AL_RGNM_TAB[R1],R5 ;POINT R5 TO REGISTER SYMBOL BLOCK
18 09 A5 0D E2 03C6 725 INCL SYMSL_VAL(R5) ;COUNT REGISTER REFERENCES
7E 05 A5 9F 03CE 726 BBS #SYMSV_CRFO,SYMSW_FLAG(R5),10\$;BRANCH IF INSERT DONE
50 04 A5 9A 03D0 727 CLRL -(SP) ;NO--CLEAR FLAGS
7E 55 50 C3 03D3 728 MOVZBL SYMSL_VAL(R5) ;POINT TO VALUE WORD
00000000'GF 04 FB 03D7 729 MOVZBL SYMSB_NAME(R5),R0 ; Get offset to symbol name/count
00000000'8F DD 03DB 730 SUBL3 R0,R5,-(SP) ; and form its address on stack
50 0000'CF 00 DD 03DF 731 PUSHAB W*MAC\$AL_CRFRCGTB ;STACK CONTROL TABLE ADDRESS
60 0060 8F B3 03E0 732 CALLS #4,G*CRF\$INSRTKEY ;INSERT REGISTER KEY
03 13 03EC 733 10\$: PUSHL #CRFSK_REF ;THIS IS A REFERENCE
6E 01 D0 03EE 734 PUSHL #0 ;ASSUME READ REFERENCE
0000'CF 9F 03EF 735 MOVL W*MAC\$GL_MOPPTR,R0 ;GET OPERAND MODE BYTE POINTER
50 04 A5 9A 03F3 736 BEQL 20\$;IF EQL NOT IN INSTRUCTION
7E 55 50 C3 03F5 737 BITW #OPDSM_MODIFY!OPDSM_WRITE,(R0) ;MODIFY OR WRITE REF?
0000'CF 9F 03FA 738 BEQL 20\$;IF EQL NO
FBEE' 30 03FC 739 MOVL #1,(SP) ;YES--SET WRITE REFERENCE
0000'CF 05 FB 03FF 740 20\$: PUSHAB W*MAC\$AB_LPBUF ;REFERENCER NAME ADDRESS
0000'CF 05 FB 0403 741 MOVZBL SYMSB_NAME(R5),R0 ; Get offset to symbol name/count
51 8ED0 0407 742 SUBL3 R0,R5,-(SP) ; and form its address on stack
05 0420 040B 743 PUSHAB W*MAC\$AL_CRFRCGTB ;CREF CONTROL TABLE ADDRESS
05 040F 744 BSBW MAC\$CVT [IN PAG ;CONVERT LINE/PAGE INTO LPBUF
05 0412 745 CALLS #5,G*CRF\$INSRTREF ;INSERT REGISTER REFERENCE
05 0419 746 INCL W*MAC\$GL_CRF_RCNT ;COUNT REGISTER REFERENCE
05 041D 747 POPL R1 ;RESTORE REGISTER NUMBER
05 0420 748 30\$: RSB


```
0421 750 .SBTTL MAC$DNUMBER ACCUMULATE DECIMAL NUMBER
0421 751
0421 752
0421 753 :++
0421 754 : FUNCTIONAL DESCRIPTION:
0421 755 :
0421 756 : This routine accumulates a decimal octaword value in
0421 757 : MAC$GO_VALUE0
0421 758 :--
0421 759 MAC$DNUMBER::
52 56 DD 0421 760 PUSHL R6 :SAVE R6
52 7C 0423 761 CLRQ R2 : Clear R2, R3, R4, R5
54 7C 0425 762 CLRQ R4 : to accumulate octaword number
52 DO AA 9E 0427 763 MOVAB -^A/0/(R10),R2 : Convert first digit to binary
042B 764 :
042B 765 : LOOP, COLLECTING THE VALUE
042B 766
042B 767 10$:
52 DD 042B 768 PUSHL R2 : Save R2
FBDO' 30 042D 769 BSBW MAC$GETCHR : Get next character
52 8ED0 0430 770 POPL R2 : Restore R2
51 DO AA 9E 0433 771 MOVAB -^A/0/(R10),R1 : Convert digit to binary
51 19 0437 772 BLSS 20$ : IF LSS THEN WE ARE DONE
09 51 91 0439 773 CMPB R1,#9 : Check if really a digit?
4C 14 043C 774 BGTR 20$ : IF GTR NO
043E 775 :
043E 776 : Multiply accumulated value by 10 and add in new digit
043E 777 :
0380 8F BB 043E 778 PUSHR #^M<R7,R8,R9> : Get some more work registers
0442 779 :
0442 780 : Multiply 1st longword by 10 and add in new digit
0442 781 :
50 51 52 0A 7A 0442 782 EMUL #10,R2,R1,R0 : Put 10*1st longword into R0,R1
03 52 1F E1 0447 783 BBC #31,R2,12$ : If sign bit set compensate for
51 0A C0 044B 784 ADDL #10,R1 : unsigned bias of 2**32
56 50 7D 044E 785 12$:
58 7C 0451 786 MOVQ R0,R6 : Accumulate result in R6,R7,R8,R9
0453 787 CLRQ R8
0453 788 :
0453 789 : Multiply 2nd longword by 10
0453 790 :
50 00 53 0A 7A 0453 791 EMUL #10,R3,#0,R0 : Put 10*2nd longword into R0,R1
03 53 1F E1 0458 792 BBC #31,R3,14$ : If sign bit set compensate for
51 0A C0 045C 793 ADDL #10,R1 : unsigned bias of 2**32
57 50 C0 045F 794 14$:
58 51 D8 0462 795 ADDL2 R0,R7 : Add value into 2nd and 3rd longwords
0465 796 ADWC R1,R8 : of octaword result
0465 797 :
0465 798 : Multiply 3rd logword by 10
0465 799 :
50 00 54 0A 7A 0465 800 EMUL #10,R4,#0,R0 : Put 10*3rd longword into R0,R1
03 54 1F E1 046A 801 BBC #31,R4,16$ : If sign bit set compensate for
51 0A C0 046E 802 ADDL #10,R1 : unsigned bias of 2**32
58 50 C0 0471 803 16$:
59 51 D8 0471 804 ADDL2 R0,R8 : Add value into 3rd and 4th longwords
0474 805 ADWC R1,R9 : of octaword result
0477 806 :
```

```

      0477 807 ; Multiply 4th longword by 10
      0477 808 ;
50    55 0A C5 0477 809 MULL3 #10,R5,R0 ; Put 10*4th longword into R0
      59 50 C0 047B 810 ADDL2 R0,R9 ; Add value into 4th longword of result
      047E 811 ;
      52 56 7D 047E 812 MOVQ R6,R2 ; Position result for next loop
      54 58 7D 0481 813 MOVQ R8,R4
      0380 8F BA 0484 814 POPR #^M<R7,R8,R9> ; Restore registers
      A1 11 0488 815 BRB 10$ ; LOOP FOR ALL DIGITS
      048A 816 ;
      048A 817 ; NUMBER IS ACCUMULATED
      048A 818 ;
      048A 819 20$:
0000'CF 52 7D 048A 820 MOVQ R2,W^MAC$GO_VALUE0 ; Store value in VALUE0
0008'CF 54 7D 048F 821 MOVQ R4,W^MAC$GO_VALUE0+8
0000'CF 53 D0 0494 822 MOVL R3,W^MAC$GL_HIGH_32 ; Set high 32 bits in case .QUAD
0000'CF 54 7D 0499 823 MOVQ R4,W^MAC$GO_HIGH_64 ; Set high 64 bits in case .OCTA
      56 8ED0 049E 824 POPL R6 ; RESTORE R6
      58 22 9A 04A1 825 MOVZBL #DINTEGER,R8 ; TOKEN IS INTEGER
      24 5A 91 04A4 826 CMPB R10,#^A/$/ ; DID WE JUST CONVERT LOCAL LABEL?
      05 12 04A7 827 BNEQ 30$ ; IF NEQ NO
      FB54' 30 04A9 828 BSBW MAC$GETCHR ; YES--SCAN PAST IT
      09 11 04AC 829 BRB MAC$GOTLOCLAB ; AND GO PROCESS THE LOCAL LABEL
      2E 5A 91 04AE 830 30$: CMPB R10,#^A/./ ; TRAILING PERIOD IN VALUE?
      03 12 04B1 831 BNEQ 40$ ; IF NEQ NO
      FB4A' 31 04B3 832 BRW MAC$GETCHR ; YES--SCAN PAST IT
      05 04B6 833 40$: RSB
```

```
04B7 835 .SBTTL MAC$GOTLOCLAB PROCESS LOCAL LABEL
04B7 836
04B7 837 :++
04B7 838 : FUNCTIONAL DESCRIPTION:
04B7 839 :
04B7 840 : THIS ROUTINE IS CALLED WHEN A LOCAL LABEL IS ENCOUNTERED.
04B7 841 :
04B7 842 : INPUTS:
04B7 843 :
04B7 844 : MAC$GQ_VALUEQ DECIMAL VALUE OF THE LOCAL LABEL
04B7 845 :--
04B7 846
04B7 847 MAC$GOTLOCLAB:
20 00 6E 00 2C 04B7 848 MOVCS #0,(SP),#0,#SYMSK_MAXLEN+1,- ;ZERO TMP SYMBOL BLOCK
0000'CF 51 0000'CF 9E 04B7 849 W^MAC$AB_TMP_SYM ;
81 06 90 04B7 850 MOVAB W^MAC$AB_TMP_SYM,R1 ;POINT TO TMP SYMBOL BLOCK
52 0000'CF D0 04C4 851 MOVB #6,(R1)+ ;NAME IS 6 CHARS LONG
81 52 D0 04C7 852 MOVL W^MAC$GL_LSB,R2 ;GET LSB NUMBER SYMBOL IS IN
0000'CF 81 0000'CF D0 04CC 853 MOVL R2,(R1)+ ;STORE AS FIRST 4 CHARS OF NAME
52 71 A0 04CF 854 MOVW W^MAC$GQ_VALUEQ,(R1)+ ;5TH AND 6TH CHARS ARE VALUE
0000'CF 52 FFFFFFFF80 8F CB 04D7 855 ADDW2 -(R1),R2 ;ACCUMULATE HASH VALUE FOR SYMBOL
FD6C 30 04E1 856 BICL3 #^C<HASHSZ>,R2,W^MAC$GL_HSHVAL ;TRIM TO HASH SIZE
09 A1 0040 8F A8 04E4 857 BSBW MAC$INSUSRSYMTB ;INSERT IN USER SYMBOL TABLE
0000'CF 51 D0 04EA 858 BISW2 #SYMSM_LOCAL,SYMSW_FLAG(R1) ;FLAG AS LOCAL LABEL
3D 5A 91 04EF 859 MOVL R1,W^MAC$GL_VALUE ;VALUE IS ADDRESS OF SYMBOL BLOCK
04 04 12 04F2 860 CMPB R10,#^A/=/ ;ASSIGNMENT OF LOCAL SYMBOL?
00 6B 0D E2 04F4 861 BNEQ 20$ ;IF NEQ NO
58 0C 9A 04F8 862 BBSS #FLGSV_OPRND,(R11),20$ ;YES--FLAG IN OPERAND FIELD
05 04FB 863 20$: MOVZBL #ID,R8 ;TOKEN TYPE IS ID
05 04FB 864 RSB
```



```
04FC 866 .SBTTL ACCUMULATE BINARY/OCTAL/HEX NUMBERS
04FC 867
04FC 868 MAC$BINNUM::
55 01 9A 04FC 869 MOVZBL #1,R5 ;SET LOG(2) OF RADIX
54 31 9A 04FF 870 MOVZBL #^A/1/,R4 ;SET UPPER CHARACTER BOUND
11 11 0502 871 BRB MAC$PNUMBER ;GO ACCUM NUMBER
0504 872
0504 873 MAC$OCTNUM::
55 03 9A 0504 874 MOVZBL #3,R5 ;SET LOG(2) OF RADIX
54 37 9A 0507 875 MOVZBL #^A/7/,R4 ;SET UPPER CHARACTER BOUND
09 11 050A 876 BRB MAC$PNUMBER
050C 877
050C 878 MAC$HEXNUM::
55 04 9A 050C 879 MOVZBL #4,R5 ;SET LOG(2) OF RADIX
54 46 8F 9A 050F 880 MOVZBL #^A/F/,R4 ;SET UPPER CHARACTER BOUND
00 11 0513 881 BRB MAC$PNUMBER
0515 882
0515 883 :++
0515 884 : FUNCTIONAL DESCRIPTION:
0515 885 :
0515 886 : THIS ROUTINE ACCUMULATES A NUMBER OF AN EVEN RADIX.
0515 887 :
0515 888 : INPUTS:
0515 889 :
0515 890 : R5 LOG(2) OF RADIX
0515 891 : R4 UPPER CHARACTER BOUND
0515 892 :
0515 893 : OUTPUTS:
0515 894 :
0515 895 : MAC$GO_VALUEO ACCUMULATED VALUE
0515 896 : R8 TOKEN VALUE
0515 897 :
0515 898 :--
0515 899
0515 900 MAC$PNUMBER::
0515 901 PUSHL R10 ;SAVE THE CURRENT CHARACTER,
0517 902 PUSHL W^MAC$GL_LINEPT ;AND LINE POINTER (IN CASE WE
051B 903 ;ACTUALLY SCAN A LOCAL SYMBOL)
51 0000'CF DD 051B 904 MOVAB W^MAC$GO_VALUEO,R1 ;Point to value octaword
61 7C 0520 905 CLRQ (R1) ;Clear result octaword
08 A1 7C 0522 906 CLRQ 8(R1)
0525 907 :
0525 908 : LOOP AND COLLECT THE VALUE
0525 909 :
50 5A D0 0525 910 10$: MOVL R10,R0 ;GET NEXT CHARACTER
30 50 91 0528 911 CMPB R0,#^A/0/ ;IS IT A DIGIT?
50 19 0528 912 BLSS 40$ ;IF LSS NO
61 8F 50 91 0528 913 CMPB R0,#^A/A/+^X20 ;IS CHARACTER LOWER-CASE ALPHA?
09 1F 0531 914 BLSSU 15$ ;IF LSSU NO
7A 8F 50 91 0533 915 CMPB R0,#^A/Z/+^X20 ;MAYBE...
03 1A 0537 916 BGTRU 15$ ;IF GTRU NOT LOWER-CASE ALPHA
50 20 8A 0539 917 BICB2 #^X20,R0 ;YES--MAKE UPPER CASE ALPHA
54 50 91 053C 918 15$: CMPB R0,R4 ;WITHIN RANGE FOR RADIX?
3C 14 053F 919 BGTR 40$ ;IF GTR NO
39 50 91 0541 920 CMPB R0,#^A/9/ ;ABOVE THE DIGITS?
0A 15 0544 921 BLEQ 20$ ;IF LEQ NO
41 8F 50 91 0546 922 CMPB R0,#^A/A/ ;YES--IS IT ABOVE LETTER 'A'?
```

```
50  F9 A0 19 054A 923      BLSS 40$      ;IF LSS NO--NUMBER IS DONE
      9E 054C 924      MOVAB -7(R0),R0      ;YES--HEX ALPHA CHARACTER--7 IS
      0550 925      ; DIFF BETWEEN 9 AND A
50  D0 A0 9E 0550 926 20$: MOVAB -^A/0/(R0),R0 ;CONVERT DIGIT TO BINARY
      53 55 D0 0554 927      MOVL R5,R3      ;SET UP LOOP COUNT
      0557 928 30$:      ;
      52 51 D0 0557 929      MOVL R1,R2      ; Reset work pointer
      82 62 C0 055A 930      ADDL2 (R2),(R2)+ ; Multiply by two,
      82 62 D8 055D 931      ADWC (R2),(R2)+ ; add high part and carry
      82 62 D8 0560 932      ADWC (R2),(R2)+ ; ...
      62 62 D8 0563 933      ADWC (R2),(R2)+ ; ...
      EE 53 F5 0566 934      SOBGTR R3,30$ ; Loop for radix
      52 51 D0 0569 935      MOVL R1,R2      ; Reset work pointer
      82 50 C0 056C 936      ADDL2 R0,(R2)+ ; Add in new digit
      82 00 D8 056F 937      ADWC #0,(R2)+ ; add carry
      82 00 D8 0572 938      ADWC #0,(R2)+ ; ...
      62 00 D8 0575 939      ADWC #0,(R2) ; ...
      FA85' 30 0578 940      BSBW MAC$GETCHR ;GET NEXT CHARACTER
      AB 11 057B 941      BRB 10$      ;CONTINUE
      057D 942      ;
      057D 943      ; NUMBER IS ACCUMULATED
      057D 944      ;
      24 5A 91 057D 945 40$: CMPB R10,#^A/8/ ;DID WE JUST SCAN A LOCAL LABEL?
      15 13 0580 946      BEQL 50$      ;IF EQL YES--GO FIX UP
      58 22 9A 0582 947      MOVZBL #DINTEGER,R8 ;NO--RETURN INTEGER TOKEN
0000'CF 0000'CF D0 0585 948      MOVL W^MAC$GL_VAL3,W^MAC$GL_HIGH_32 ;SAVE HIGH 32 BITS
0000'CF 0000'CF 7D 058C 949      MOVQ W^MAC$GQ_VAL2,W^MAC$GQ_HIGH_64 ; Save high 64 bits
      SE 08 C0 0593 950      ADDL2 #2*4,SP ;CLEAN STACK
      05 0596 951      RSB
      0597 952      ;
      0597 953      ; IT APPEARS THAT WE ACTUALLY SCANNED A LOCAL SYMBOL. THIS CAN HAPPEN
      0597 954      ; WHEN SOMETHING LIKE '^A0<12+47$+12>' OCCURS.
      0597 955      ;
0000'CF 8ED0 0597 956 50$: POPL W^MAC$GL_LINEPT ;RESTORE LINE POINTER
      SA 8ED0 059C 957      POPL R10 ;RESTORE CHARACTER
      FE7F 31 059F 958      BRW MAC$DNUMBER ;GO RESCAN AS DECIMAL NUMBER
      05A2 959
      05A2 960      .END
```

MACSSCANER
Symbol table

SCANNING ROUTINES

F 10

16-SEP-1984 02:14:19
5-SEP-1984 01:49:51

VAX/VMS Macro V04-00
[MACRO.SRC]SCANNER.MAR;1

Page 23
(13)

SCOUNT = 0000003B
AB = 00000001
AD = 0000C008
AF = 00008004
AG = 0000A008
AH = 00009010
AL = 00000004
AO = 00000010
AQ = 00000008
ARGSK_SIZE = 000003E8
AUDSK_SIZE = 00000010
AW = 00000002
B = 00000001
BLNK = 00000020
CHRSM_COMMA_CR = 00000020
CHRSM_ILL_CHR = 00000040
CHRSM_NUM_BER = 00000010
CHRSM_SPA_MSK = 00000001
CHRSM_SYM_CH1 = 00000008
CHRSM_SYM_CHR = 00000004
CHRSM_SYM_DLM = 00000002
CHRSV_COMMA_CR = 00000005
CHRSV_CVTLWC = 00000061
CHRSV_ILL_CHR = 00000006
CHRSV_NOCVT = 0000007F
CHRSV_NUM_BER = 00000004
CHRSV_SPA_MSK = 00000000
CHRSV_SYM_CH1 = 00000003
CHRSV_SYM_CHR = 00000002
CHRSV_SYM_DLM = 00000001
CNT = 00000002
CR = 0000000D
CRFSINSRTKEY *****
CRFSINSRTREF *****
CRFSK_REF *****
CRFSM_DEFAULT = 00000012
CRFSM_DIR = 00000001
CRFSM_MACROS = 00000002
CRFSM_OPCODES = 00000004
CRFSM_REGISTERS = 00000008
CRFSM_SYMBOLS = 00000010
CRFSV_DIR = 00000000
CRFSV_MACROS = 00000001
CRFSV_OPCODES = 00000002
CRFSV_REGISTERS = 00000003
CRFSV_SYMBOLS = 00000004
D = 0000C008
DAND = 0000001D
DANGCLS = 00000016
DANGOPN = 00000015
DAT = 00000020
DBUP = 0000002B
DCLS = 00000018
DCOLON = 00000010
DCOMMA = 0000000F
DDIV = 0000001C
DEOL = 0000000B

X 05
X 05
X 05

DEQ = 00000011
DGUP = 0000002C
DINTEGER = 00000022
DIUP = 0000002D
DLUP = 0000002E
DMASK = 00000032
DMINUS = 0000001A
DOPCODE = 0000000E
DOPN = 00000017
DOR = 0000001E
DPC = 00000012
DPLUS = 00000019
DPOUND = 00000021
DSQCLS = 00000014
DSQOPN = 00000013
DSUP = 0000002F
DTIMES = 0000001B
DUPA = 00000023
DUPB = 00000024
DUPC = 00000025
DUPD = 00000026
DUPF = 00000028
DUPM = 00000029
DUPO = 00000027
DUPX = 0000002A
DWUP = 00000030
DXOR = 0000001F
ERR = 00000001
ERR01 = 00000001
ERR02 = 00000002
ERR03 = 00000003
ERR04 = 00000004
ERR05 = 00000005
ERR06 = 00000006
ERR07 = 00000007
ERR08 = 00000008
ERR09 = 00000009
F = 00008004
FF = 0000000C
FLGSM_ALLCHR = 00000001
FLGSM_BOL = 00000002
FLGSM_CHKLPND = 00100000
FLGSM_COMPEXPR = 00000004
FLGSM_CONT = 00000008
FLGSM_CRF = 40000000
FLGSM_CRSEEN = 00000001
FLGSM_DATRPT = 00000010
FLGSM_DBGOUT = 00004000
FLGSM_DLMSTR = 00008000
FLGSM_ENDMCH = 00000020
FLGSM_EVAEXPR = 00000040
FLGSM_EXOPT = 00000080
FLGSM_EXTERR = 00010000
FLGSM_EXTWRN = 00020000
FLGSM_FIRSTLN = 00000200
FLGSM_IFSTAT = 00800000
FLGSM_IIF = 00400000

FLGSM_INSERT = 00000100
FLGSM_IRPC = 20000000
FLGSM_LEXOP = 00000002
FLGSM_LSTXST = 00000200
FLGSM_MAC2COL = 00000800
FLGSM_MACL = 00000800
FLGSM_MACLTB = 08000000
FLGSM_MACTXT = 00010000
FLGSM_MEBLST = 00001000
FLGSM_MOREARG = 00002000
FLGSM_MOREINP = 00000008
FLGSM_NEWPND = 00000400
FLGSM_NOREF = 01000000
FLGSM_NTTYPEPC = 00000020
FLGSM_NULCHR = 00040000
FLGSM_OBJXST = 00200000
FLGSM_OPNDCHK = 00000100
FLGSM_OPRND = 00002000
FLGSM_OPTVFLIDX = 00001000
FLGSM_ORDLST = 00020000
FLGSM_P2 = 00004000
FLGSM_RPTIRP = 10000000
FLGSM_SEQFIL = 02000000
FLGSM_SKAN = 00008000
FLGSM_SPECOP = 00000004
FLGSM_SPLALL = 04000000
FLGSM_STOIMF = 00040000
FLGSM_SYM2COL = 00000400
FLGSM_TOCFILG = 00080000
FLGSM_UPAFILG = 00000010
FLGSM_UPDFIL = 00000080
FLGSM_UPMARG = 00000040
FLGSM_XCRF = 80000000
FLGSV_ALLCHR = 00000000
FLGSV_BOL = 00000001
FLGSV_CHKLPND = 00000014
FLGSV_COMPEXPR = 00000002
FLGSV_CONT = 00000003
FLGSV_CRF = 0000001E
FLGSV_CRSEEN = 00000020
FLGSV_DATRPT = 00000004
FLGSV_DBGOUT = 0000002E
FLGSV_DLMSTR = 0000002F
FLGSV_ENDMCH = 00000005
FLGSV_EVAEXPR = 00000006
FLGSV_EXOPT = 00000007
FLGSV_EXTERR = 00000030
FLGSV_EXTWRN = 00000031
FLGSV_FIRSTLN = 00000029
FLGSV_IFSTAT = 00000017
FLGSV_IIF = 00000016
FLGSV_INSERT = 00000008
FLGSV_IRPC = 0000001D
FLGSV_LEXOP = 00000021
FLGSV_LSTXST = 00000009
FLGSV_MAC2COL = 0000002B
FLGSV_MACL = 0000000B

MACSSCANER
Symbol table

SCANNING ROUTINES

G 10

16-SEP-1984 02:14:19
5-SEP-1984 01:49:51

VAX/VMS Macro V04-00
[MACRO.SRC]SCANNER.MAR;1

Page 24
(13)

FLGSV_MACTB = 0000001B
FLGSV_MACTXT = 00000010
FLGSV_MEBLST = 0000000C
FLGSV_MOREARG = 0000002D
FLGSV_MOREINP = 00000023
FLGSV_NEWPND = 0000000A
FLGSV_NOREF = 00000018
FLGSV_NTTYPEPC = 00000025
FLGSV_NULCHR = 00000032
FLGSV_OBXST = 00000015
FLGSV_OPNDCHK = 00000028
FLGSV_OPRND = 0000000D
FLGSV_OPTVFLIDX = 0000002C
FLGSV_ORDLST = 00000011
FLGSV_P2 = 0000000E
FLGSV_RPTIRP = 0000001C
FLGSV_SEQFIL = 00000019
FLGSV_SKAN = 0000000F
FLGSV_SPECOP = 00000022
FLGSV_SPLALL = 0000001A
FLGSV_STOIMF = 00000012
FLGSV_SYM2COL = 0000002A
FLGSV_TOCLFG = 00000013
FLGSV_UPAFILG = 00000024
FLGSV_UPDFIL = 00000027
FLGSV_UPMARG = 00000026
FLGSV_XCRF = 0000001F
G = 0000A008
GOALSY = 0000000A
H = 00009010
HASHSZ = 0000007F
HYPHEN = 0000002D
ID = 0000000C
INPSK_BUFSIZ = 000003E8
INSYM = 00000002
INSYMP = 000001F7
INSYTM = 000001F7
INTSK_BUFSIZ = 000013F4
INTSK_BUFWRN = 00001390
INTS_ADD = 00000001
INTS_AND = 00000002
INTS_ASH = 00000003
INTS_ASN = 0000000C
INTS_AUGPC = 0000000D
INTS_BDST = 0000000E
INTS_CHKL = 0000000F
INTS_DIV = 00000004
INTS_END = 00000010
INTS_EPT = 00000011
INTS_ERR = 00000012
INTS_ETX = 00000013
INTS_FNEWL = 00000014
INTS_ILG = 00000000
INTS_INFO = 0000003A
INTS_LGLAB = 00000015
INTS_MACL = 00000016
INTS_MUL = 00000005

R 04
R 04

INTS_NEG = 00000006
INTS_NEWL = 00000017
INTS_NEWP = 00000018
INTS_NOT = 00000007
INTS_OP = 00000019
INTS_OR = 00000008
INTS_PRIL = 0000001A
INTS_PRT = 0000001B
INTS_PSECT = 0000001C
INTS_REDEF = 0000001D
INTS_REF = 0000001E
INTS_REST = 0000001F
INTS_SAME = 00000009
INTS_SAVE = 00000020
INTS_SBTTL = 00000021
INTS_SETFLAG = 00000022
INTS_SETLONG = 00000023
INTS_SPIC = 00000024
INTS_SPID = 00000025
INTS_STIB = 00000026
INTS_STIL = 00000028
INTS_STIW = 00000027
INTS_STKEPT = 00000029
INTS_STKG = 0000002A
INTS_STKL = 0000002B
INTS_STKPC = 0000002C
INTS_STKS = 0000002D
INTS_STOB = 00000034
INTS_STOL = 0000002E
INTS_STOW = 00000035
INTS_STRB = 0000002F
INTS_STRL = 00000031
INTS_STRSB = 00000032
INTS_STRSW = 00000033
INTS_STRW = 00000030
INTS_STSB = 00000036
INTS_STSW = 00000037
INTS_SUB = 0000000A
INTS_SUME = 00000039
INTS_WRN = 00000038
INTS_XOR = 0000000B
KADDRESS = 00000037
KALIGN = 0000005A
KASCIC = 00000033
KASCID = 00000078
KASCII = 00000034
KASCIZ = 00000035
KBLKA = 0000003F
KBLKB = 00000040
KBLKD = 00000041
KBLKF = 00000042
KBLKG = 0000007E
KBLKH = 0000007F
KBLKL = 00000043
KBLKO = 00000080
KBLKQ = 00000044
KBLKW = 00000045

KBYTE = 00000038
KCROSS = 00000079
KDEBUG = 00000055
KDFLT = 00000078
KDOUBLE = 00000039
KDSABL = 00000056
KENABL = 00000057
KEND = 00000076
KENDC = 0000004E
KENDM = 00000053
KENDR = 0000004F
KENTRY = 00000058
KERROR = 00000071
KEVEN = 0000005B
KEXTRN = 0000005D
KFIELD = 0000003A
KFLOAT = 0000003B
KGFLOAT = 00000081
KGLOBAL = 0000005E
KHFLOAT = 00000082
KIDENT = 0000006A
KIF = 00000046
KIFF = 00000048
KIFT = 00000049
KIFTF = 0000004A
KIIF = 00000047
KINCLUDE = 0000005F
KIRP = 0000004B
KIRPC = 0000004C
KLIBRARY = 00000060
KLINK = 00000085
KLIST = 00000061
KLONG = 0000003C
KMACRO = 00000050
KMCALL = 00000051
KMDELETE = 00000054
KMEXIT = 00000052
KNARG = 00000063
KNCHR = 00000064
KNCROS = 0000007A
KNLIST = 00000062
KNTYPE = 00000074
KOCTA = 00000083
KODD = 0000005C
KOPDEF = 00000075
KPACKED = 00000036
KPAGE = 00000065
KPRINT = 00000072
KPSECT = 00000066
KQUAD = 0000003D
KREF1 = 0000006D
KREF16 = 00000084
KREF2 = 0000006E
KREF4 = 0000006F
KREF8 = 00000070
KREPT = 0000004D
KRESTORE = 00000067

MAC\$SCANER
Symbol table

SCANNING ROUTINES

H 10

16-SEP-1984 02:14:19
5-SEP-1984 01:49:51

VAX/VMS Macro V04-00
[MACRO.SRC]SCANER.MAR;1

Page 25
(13)

KSAVE = 00000068
KSBTTL = 0000006B
KSGNB = 0000007C
KSGNW = 0000007D
KTITLE = 00000069
KVECTOR = 00000059
KWARN = 00000073
KWEAK = 0000006C
KWORD = 0000003E
KXFER = 00000077
L = 00000004
LENSK_UPXTAB ***** X 05
LENSK_XUPTAB ***** X 05
LIB\$GET_VM ***** X 05
LST\$K_BUF\$SIZ = 00000086
LST\$K_L_P_PAGE = 0000003C
LST\$K_TITLE_SIZ = 00000028
MAC\$AB_CMSK_TAB ***** X 05
MAC\$AB_LPBUF ***** X 05
MAC\$AB_TMP\$Y1 ***** X 05
MAC\$AB_TMP\$Y2 ***** X 05
MAC\$AB_UPXTAB ***** X 05
MAC\$AB_UPXTOKEN ***** X 05
MAC\$AB_XUPTAB ***** X 05
MAC\$AB_XUPTOKEN ***** X 05
MAC\$AL_CRFRGCTB ***** X 05
MAC\$AL_PRMHSHTB ***** X 05
MAC\$AL_RGNM_TAB 00000030 RG 03
MAC\$AL_UMCHSHTB ***** X 05
MAC\$AL_USYHSHTB ***** X 05
MAC\$BINNUM 000004FC RG 05
MAC\$CHKREG 00000075 R 05
MAC\$CHRRER ***** X 05
MAC\$CREF_DIR ***** X 05
MAC\$CVT_LIN_PAG ***** X 05
MAC\$CVT_SYM_UP 0000013F RG 05
MAC\$DNUMBER 00000421 RG 05
MAC\$ERR_NOMEM ***** X 05
MAC\$GB_RDXNDX ***** X 05
MAC\$GETCHR ***** X 05
MAC\$GETDOUBLE ***** X 05
MAC\$GETFLOAT ***** X 05
MAC\$GETGFLOAT ***** X 05
MAC\$GETHFLOAT ***** X 05
MAC\$GETSYM 00000171 RG 05
MAC\$GL_BASEADDR ***** X 05
MAC\$GL_CRF_FLG ***** X 05
MAC\$GL_CRF_RCNT ***** X 05
MAC\$GL_HIGH_32 ***** X 05
MAC\$GL_HSHVAL ***** X 05
MAC\$GL_LINEPT ***** X 05
MAC\$GL_LSB ***** X 05
MAC\$GL_MCLVL ***** X 05
MAC\$GL_MOPPTR ***** X 05
MAC\$GL_OPCLSTPT ***** X 05
MAC\$GL_PSECT ***** X 05
MAC\$GL_SYMPGPTR ***** X 05

MAC\$GL_SYMPGREQ ***** X 05
MAC\$GL_SYM_PAGL ***** X 05
MAC\$GL_VAL3 ***** X 05
MAC\$GL_VALUE ***** X 05
MAC\$GOTLOCLAB 000004B7 R 05
MAC\$GO_VALUE0 ***** X 05
MAC\$GQ_HIGH_64 ***** X 05
MAC\$GQ_VAL2 ***** X 05
MAC\$GQ_VALUEQ ***** X 05
MAC\$HEXNUM 0000050C RG 05
MAC\$IMPLMCALL ***** X 05
MAC\$INSUSRSYMTB 00000250 RG 05
MAC\$INTERR_2_LW ***** X 05
MAC\$LCCLSKIP 0000011E RG 05
MAC\$NUMBER 000000C5 RG 05
MAC\$OCTNUM 00000504 RG 05
MAC\$PNUMBER 00000515 RG 05
MAC\$REGIS 00000354 R 05
MAC\$SKIPSP ***** X 05
MAC\$SRC\$SYMTAB 00000247 RG 05
MAC\$SRCUSRSYMTB 00000256 RG 05
MAC\$SRC_LIST 0000023A RG 05
MAC\$SYMBOL 00000000 RG 05
MAC\$SYMNUM 00000083 RG 05
MAC\$SYM\$CN 0000010C RG 05
MAC\$SYM\$CNUP 000000F2 RG 05
MAC\$SYM\$CNUP1 000000F9 RG 05
MAC\$XPOUND 000001FC RG 05
MAC\$XSYMBL 00000099 RG 05
MAC\$XUPARROW 000001D7 RG 05
MAC\$ILLSYMLEN = 007D8820
MAC\$XTXT = 0000000D
MAC_SUBSYS = 0000007D
MB = 00000041
MD = 0000C048
MF = 00008044
MG = 0000A048
MH = 00009050
ML = 00000044
MO = 00000050
MQ = 00000048
MW = 00000042
NUM_TAB 000000D2 R 05
OBJ\$K_BUF\$SIZ = 00000200
OPD\$M_ADDR = 00000000
OPD\$M_BB = 000000A1
OPD\$M_BW = 000000C2
OPD\$M_D_FLOAT = 0000C000
OPD\$M_FLOAT = 00008000
OPD\$M_G_FLOAT = 0000A000
OPD\$M_H_FLOAT = 00009000
OPD\$M_MODE = 000003E0
OPD\$M_MODIFY = 00000040
OPD\$M_NOT_32F = 00007000
OPD\$M_READ = 00000020
OPD\$M_VIELD = 00000080

OPD\$M_WRITE = 00000060
OPD\$M_MODE = 00000005
OPD\$M_SIZE = 00000005
OPD\$M_D_FLOAT = 0000000E
OPD\$M_FLOAT = 0000000F
OPD\$M_G_FLOAT = 0000000D
OPD\$M_H_FLOAT = 0000000C
OPD\$M_MODE = 00000005
OPD\$M_SIZE = 00000000
OPF\$M_LASTOPR = 00002000
OPF\$M_OPTEXP = 00001000
OPF\$V_LASTOPR = 0000000D
OPF\$V_OPTEXP = 0000000C
PSC\$B_NAME 00000004
PSC\$B_SEG 0000000C
PSC\$B_UNUSED 0000000B
PSC\$K_BLK\$SIZ 00000013
PSC\$K_NO_OPTNS = 0000000A
PSC\$LCURLOC 0000000F
PSC\$LINK 00000000
PSC\$MAXLGTH 00000005
PSC\$M_ABS = FFFFFFFF7
PSC\$M_ALIGNFLG = 00004000
PSC\$M_ALLOPTNS = 000003FF
PSC\$M_BYTE = 00004000
PSC\$M_CON = FFFFFFFFB
PSC\$M_DEFAULT = 000001C8
PSC\$M_EXE = 000000C0
PSC\$M_GBL = 00000010
PSC\$M_LCL = FFFFFFFEF
PSC\$M_LIB = 00000002
PSC\$M_LONG = 00004800
PSC\$M_NOEXE = FFFFFFFBF
PSC\$M_NOPIC = FFFFFFFFE
PSC\$M_NORD = FFFFFFFF7F
PSC\$M_NOSHR = FFFFFFFDF
PSC\$M_NOVEC = FFFFFFFDF
PSC\$M_NOWRT = FFFFFFFEF
PSC\$M_OVR = 00000004
PSC\$M_PAGE = 00006400
PSC\$M_PIC = 00000001
PSC\$M_QUAD = 00004C00
PSC\$M_RD = 00000080
PSC\$M_REL = 00000008
PSC\$M_SHR = 00000020
PSC\$M_USR = FFFFFFFFD
PSC\$M_VEC = 00000200
PSC\$M_WORD = 00004400
PSC\$M_WRT = 00000180
PSC\$S_ALIGNMENT = 00000004
PSC\$V_ALIGNFLG = 0000000E
PSC\$V_ALIGNMENT = 0000000A
PSC\$V_EXE = 00000006
PSC\$V_GBL = 00000004
PSC\$V_LIB = 00000001
PSC\$V_OVR = 00000002
PSC\$V_PIC = 00000000

MAC\$SCANER
Symbol table

SCANNING ROUTINES

I 10

16-SEP-1984 02:14:19 VAX/VMS Macro V04-00
5-SEP-1984 01:49:51 [MACRO.SRC]SCANER.MAR;1

Page 26
(13)

PSC\$V_RD = 00000007
PSC\$V_REL = 00000003
PSC\$V_SHR = 00000005
PSC\$V_VEC = 00000009
PSC\$V_WRT = 00000008
PSC\$W_FLAG = 00000009
PSC\$W_OPTIONS = 0000000D
Q = 00000008
RB = 00000021
RD = 0000C028
RDX\$V_BINARY = 00000000
RDX\$V_DECIMAL = 00000002
RDX\$V_DOUBLE = 00000005
RDX\$V_FLOAT = 00000004
RDX\$V_GFLOAT = 00000006
RDX\$V_HEX = 00000003
RDX\$V_HFLOAT = 00000007
RDX\$V_OCTAL = 00000001
REG\$ PC = 0000000F
REG_CREF = 000003B2 R 05
REG_SYMB_AP = 00000187 RG 04
REG_SYMB_FP = 000001A7 RG 04
REG_SYMB_PC = 000001E7 RG 04
REG_SYMB_R0 = 00000003 RG 04
REG_SYMB_R1 = 00000023 RG 04
REG_SYMB_R10 = 00000144 RG 04
REG_SYMB_R11 = 00000166 RG 04
REG_SYMB_R2 = 00000043 RG 04
REG_SYMB_R3 = 00000063 RG 04
REG_SYMB_R4 = 00000083 RG 04
REG_SYMB_R5 = 000000A3 RG 04
REG_SYMB_R6 = 000000C3 RG 04
REG_SYMB_R7 = 000000E3 RG 04
REG_SYMB_R8 = 00000103 RG 04
REG_SYMB_R9 = 00000123 RG 04
REG_SYMB_SP = 000001C7 RG 04
RF = 00008024
RG = 0000A028
RH = 00009030
RL = 00000024
RO = 00000030
RQ = 00000028
RRREG = 00000031
RW = 00000022
SEMI = 0000003B
STB\$K_PG_MISS = 0000000A
SYMSB_NAME = 00000004
SYMSB_SEG = 0000000C
SYMSB_TOKEN = 0000000B
SYMSK_BLKSIZE = 0000000D
SYMSK_MAXLEN = 0000001F
SYMSK_TWOCOL = 00000010
SYMSL_LINK = 00000000
SYMSL_VAL = 00000005
SYMSM_ABS = 00000010
SYMSM_ASN = 00000100
SYMSM_CRFO = 00002000

SYMSM_DEBUG = 00000020
SYMSM_DEF = 00000001
SYMSM_DELMAC = 00000200
SYMSM_EPT = 00000200
SYMSM_EXTRN = 00000008
SYMSM_GLOBL = 00000004
SYMSM_LOCAL = 00000040
SYMSM_ODBG = 00000400
SYMSM_REF = 00000080
SYMSM_RELPSECT = 00000800
SYMSM_SUPR = 00004000
SYMSM_WEAK = 00000002
SYMSM_XCRF = 00001000
SYMSV_ABS = 00000004
SYMSV_ASN = 00000008
SYMSV_CRFO = 0000000D
SYMSV_DEBUG = 00000005
SYMSV_DEF = 00000000
SYMSV_DELMAC = 00000009
SYMSV_EPT = 00000009
SYMSV_EXTRN = 00000003
SYMSV_GLOBL = 00000002
SYMSV_LOCAL = 00000006
SYMSV_ODBG = 0000000A
SYMSV_REF = 00000007
SYMSV_RELPSECT = 0000000B
SYMSV_SUPR = 0000000E
SYMSV_WEAK = 00000001
SYMSV_XCRF = 0000000C
SYMSW_FLAG = 00000009
TAB = 00000009
TWO_CHR_REG_NAM = 00000000 R 03
TWO_CHR_REG_SIZE = 00000020
TWO_CHR_REG_VAL = 00000020 R 03
VB = 00000081
VD = 0000C088
VF = 00008084
VG = 0000A088
VH = 00009090
VL = 00000084
VO = 00000090
VQ = 00000088
VW = 00000082
W = 00000002
WB = 00000061
WD = 0000C068
WF = 00008064
WG = 0000A068
WH = 00009070
WL = 00000064
WO = 00000070
WQ = 00000068
WW = 00000062
X1 = 00000033
X2 = 00080000

+-----+
! Psect synopsis !
+-----+

| PSECT name | Allocation | PSECT No. | Attributes |
|-----------------|-------------------|-----------|---------------------------------------------------------|
| . ABS : | 00000000 (0.) | 00 (0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| . BLANK : | 00000000 (0.) | 01 (1.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| \$ABSS | 00000013 (19.) | 02 (2.) | NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE |
| MAC\$RO_DATA | 00000070 (112.) | 03 (3.) | NOPIC USR CON REL GBL NOSHR NOEXE RD NOWRT NOVEC LONG |
| MAC\$RW_DATA | 00000204 (516.) | 04 (4.) | NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG |
| MAC\$RO_CODE_P1 | 000005A2 (1442.) | 05 (5.) | NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG |

+-----+
! Performance indicators !
+-----+

| Phase | Page faults | CPU Time | Elapsed Time |
|------------------------|-------------|-------------|--------------|
| Initialization | 29 | 00:00:00.04 | 00:00:00.88 |
| Command processing | 125 | 00:00:00.41 | 00:00:03.56 |
| Pass 1 | 276 | 00:00:05.37 | 00:00:20.71 |
| Symbol table sort | 0 | 00:00:00.71 | 00:00:02.33 |
| Pass 2 | 187 | 00:00:01.72 | 00:00:07.89 |
| Symbol table output | 51 | 00:00:00.26 | 00:00:00.62 |
| Psect synopsis output | 2 | 00:00:00.02 | 00:00:00.02 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 672 | 00:00:08.53 | 00:00:36.32 |

The working set limit was 1500 pages.
53269 bytes (105 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 704 non-local and 75 local symbols.
960 source lines were read in Pass 1, producing 29 object records in Pass 2.
17 pages of virtual memory were used to define 13 macros.

+-----+
! Macro library statistics !
+-----+

| Macro library name | Macros defined |
|--------------------------------------|----------------|
| _\$255\$DUA28:[MACRO.OBJ]MACRO.MLB;1 | 12 |
| -\$255\$DUA28:[SYSLIB]STARLET.MLB;2 | 3 |
| TOTALS (all libraries) | 15 |

739 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SCANER/OBJ=OBJ\$:SCANER MSRC\$:SCANER/UPDATE=(ENH\$:SCANER)+LIB\$:MACRO/LIB

0227 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

